New Factorization Techniques **and Parallel** $O(\text{Log } N)$ **Algorithms for Forward Dynamics Solution of Single Closed-Chain Robot Manipulators**

Amir Fijany

Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109

Abstract- In this paper parallel $O(\text{Log } N)$ algorithms for dynamic simulation of single closed-chain rigid multibody system as specialized to the case of a robot manipulator in contact with the environment are developed. A new factorization technique is proposed for computation of the inverse of the mass matrix ($\mathcal{M}^{-1}$) and the inverse of the Operational Space Mass Matrix (Λ^{-1}). This results in new factorization for both $\mathcal{M}^{-1}$ and Λ^{-1} in form of *Schur* Complement with simple physical interpretations. A new O(N) algorithm for the problem is then derived by a recursive implementation of these factorization, It is shown that this O(N) algorithm is ***strictlyparallel***, that is, it is less efficient than other O(N) algorithms for serial computation of problem. But, to our knowledge, it is the only known algorithm that can be parallelized and lead to a both time and processor-optimal parallel algorithms for the problem, i.e., parallel $O(\log N)$ algorithms with O(N) processors. The parallel algorithms developed in this paper, in addition to their theoretical significance, are also practical from an implementation point of view due to their simple architectural requirements.

NOMENCLATURE

N	Number of total Degree-Of-Freedom (DOF) of system
$P_{i,j}$	Position vector from O_j to O_i, $P_{i+1\,i} = p_i$
$O(i, i+1)$	3x3 matrix describing the orientation of frame i+1 with respect to frame i.
m_i	Mass of link i
h_i, k_i	First and Second Moment of mass of link i about point O_i
I_i	Spatial inertia of link i about point O_i
$\mathcal{I} \triangleq \mathrm{diag}\{I_i\} \varepsilon \mathbb{R}^{6N\times 6N}$	Global matrix of spatial inertias, i = N to 1
$\mathcal{M} \varepsilon \mathbb{R}^{N\times N}$	Symmetric Positive Definite (SPD) mass matrix
$\Lambda^{-1} \varepsilon \mathbb{R}^{6\times 6}$	Inverse of Operational Space mass matrix
$\mathcal{J} \varepsilon \mathbb{R}^{6\times N}$	Jacobian matrix
$\theta \triangleq \mathrm{col}\{\theta_i\} \varepsilon \mathbb{R}^{N\times 1}$	Global vector of joint positions, i = N to 1
$Q \triangleq \mathrm{col}\{Q_i\} \varepsilon \mathbb{R}^{N\times 1}$	Global vector of joint velocities, i = N to 1
$\dot{Q} \triangleq \mathrm{col}\{\dot{Q}_i\} \varepsilon \mathbb{R}^{N\times 1}$	Global vector of joint accelerations, i = N to 1
$\mathcal{T} \triangleq \mathrm{col}\{\tau_i\} \varepsilon \mathbb{R}^{N\times 1}$	Global vector of applied joint forces, i = N to 1
w_i, $\dot{\omega}_i \varepsilon \mathbb{R}^{3\times 1}$	Angular and linear acceleration of link i (frame i+1)
v_i, $\dot{v}_i \varepsilon \mathbb{R}^{3\times 1}$	Linear velocity and acceleration of link i (point O_i)
$V_i \triangleq \begin{bmatrix} \omega_i \\ v_i \end{bmatrix} \varepsilon \mathbb{R}^{6\times 1}$	Spatial velocity of link i
$\dot{V}_i \triangleq \begin{bmatrix} \dot{\omega}_i \\ \dot{v}_i \end{bmatrix} \varepsilon \mathbb{R}^{6\times 1}$	Spatial acceleration of link i
$\mathcal{V} \triangleq \mathrm{col}\{V_i\} \varepsilon \mathbb{R}^{6N\times 1}$	Global vector of link Velocities, i = N to 1
$\dot{\mathcal{V}} \triangleq \mathrm{col}\{\dot{V}_i\} \varepsilon \mathbb{R}^{6N\times 1}$	Global vector of link accelerations, i = N to 1
f_i, $n_i \varepsilon \mathbb{R}^{3\times 1}$	Force and moment of interaction between link i-1 and link i
$F_i \triangleq \begin{bmatrix} n_i \\ f_i \end{bmatrix} \varepsilon \mathbb{R}^{6\times 1}$	Spatial force of interaction between link i-1 and link i
$\mathcal{F} \triangleq \mathrm{col}\{F_i\} \varepsilon \mathbb{R}^{6N\times 1}$	Global vector of interaction forces, i = N to 1
$H_i \varepsilon \mathbb{R}^{6\times 1}$	Spatial axis (map matrix) of joint i
$\mathcal{H} \triangleq \mathrm{diag}\{H_i\} \varepsilon \mathbb{R}^{6N\times 6N}$	Global matrix of spatial axes, i = N to 1

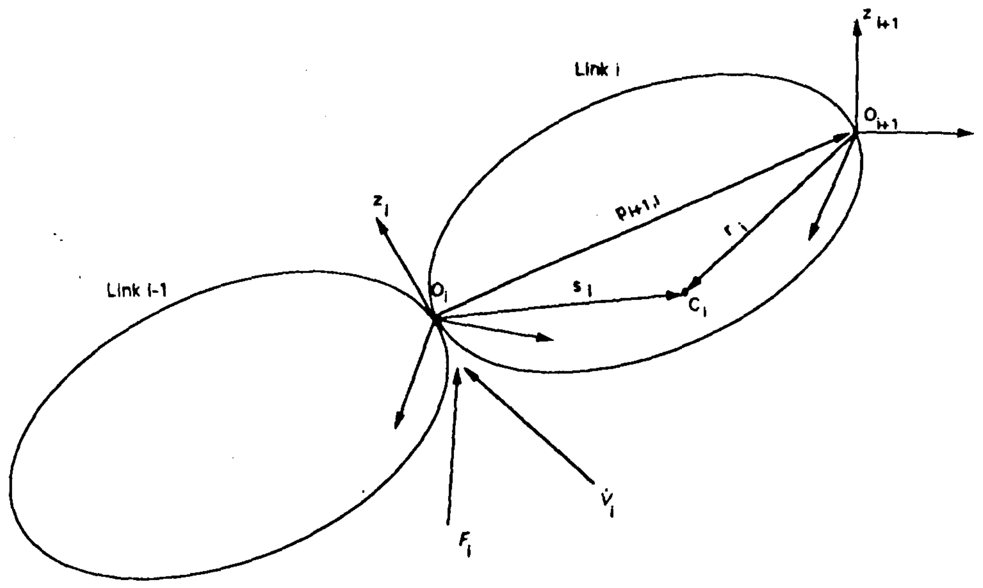

Figure I. Links, Frames, and Position Vectors
CI : Center of Mass of Link i

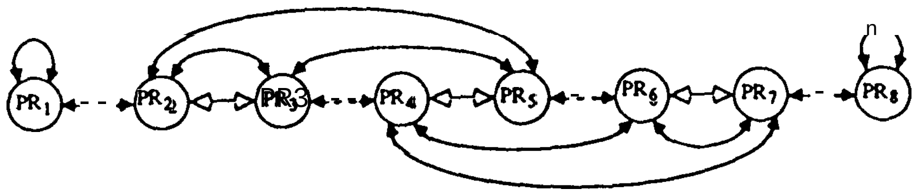

Figure 2. Perfect Shuffle Exchange augmented with Nearest Neighbor (SENN) Interconnection,
Solid lines are shuffle and dashed lines are exchange.
Double solid lines convert the interconnection to nearest neighbor.

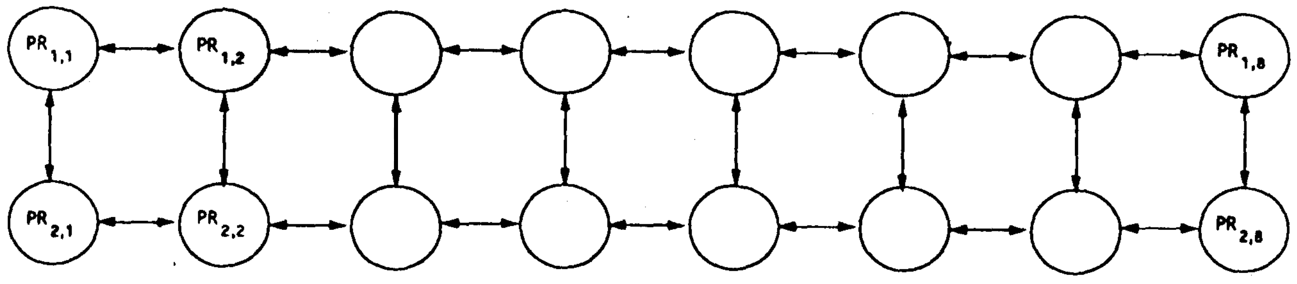

Figure 3. Architecture for Multilevel Parallel Computation of the Algorithm. (Shuffle Exchange horizontal processors interconnection is not shown)

I. INTRODUCTION

The application of single open-chain robot manipulators for many typical tasks involves the interaction with the environment, This interaction constrains the motion of the End-Effecter (EE), resulting in a single closed-chain system, Although, topologically, such a system represents a rather simple example of closed-chain mechanisms, from a computational point of view, its dynamics simulation (or, forward dynamics solution) is significantly more demanding than that of a single open-chain system. In addition to the solution for constraint force, it requires twice the solution of forward dynamics of the open-chain system as well as the computation of the inverse of the Operational Space Mass Matrix [II of the open-chain system, Λ^{-1}.

Serial algorithms for forward dynamics of single closed-chain system have been proposed by Orin and McGhee [2], Featherstone [3,4], Lathrop [5], Lilly [61, Lilly and Orin [7,81, Brandle, Johanni, and Otter [91, Rodriguez, Kreutz-Delgado, and Jain [10]. The works in [5,6,8,9,101 include the development of O(N) algorithms which, asymptotically, represent the most efficient method for serial solution of the problem.

Despite the significant improvement in the efficiency of serial algorithms; there is considerable motivation for devising faster algorithms for the problem. This ranges from the need for extensive off-line simulation capability for design and evaluation purposes to real-time implementation for controlled simulations and teleoperator training. In particular, for many anticipated space teleoperation application a faster-than-real-time simulation capability will be essential [11,12]. Given the relative maturity of serial algorithms, it is clear that any significant improvement in the computational efficiency can be only achieved through exploitation of a high degree of parallelism. However, unlike the serial computation, there seems to be no report on the development of parallel algorithms for the problem.

The development of efficient parallel algorithms for forward dynamics of both single open- and closed-chain systems is a rather challenging problem. It represents an interesting example for which the analysis of the efficiency of a given algorithm for parallel computation is far different and more complex than that for serial computation. In fact, the previous results [12,13,14] and the results of this paper clearly indicate that those algorithms that are less efficient (in terms of either asymptotic complexity or number of operations) for serial computation provide a higher degree of parallelism and hence are more efficient for parallel computation.

From an algorithmic standpoint, the forward dynamics solution of single open- and closed-chain systems are closely related. The O(N) algorithms for the single closed-chain system rely on an O(N) procedure for solution of the open-chain system. They also rely on an O(N) recursive procedure for computation of Λ^{-1}. This recursive procedure has a structure similar to that encountered in the computation of the O(N) algorithms for open-chain systems, or more precisely, in the computation of the articulated-body inertia (for' example, compare Eq. (43) in [81 with Eq. (45) in [15].) Consequently, the challenge in parallelization of the O(N) algorithms for both single open- and closed-chain systems is the same. More precisely, it resides in the parallelization of the structurally similar nonlinear recurrences that arise in the computation of the articulated-body inertia and Λ^{-1}.

An investigation of parallelism in forward dynamics solution of open-chain systems by analyzing the efficiency of existing algorithms for parallel computation is reported in [12,13]. This investigation rely on two fundamental results, established in [16], regarding the O(N) algorithms for the problem. The first result is that, at a conceptual level, the O(N) algorithms can be essentially considered as a procedure for recursive factorization and inversion of mass matrix [17,18]. The second result is that, at a computational level, the O(N) algor thins lead to the computation of the articulated-body inertia [15].

Building upon these two fundamental results, the investigation in [12,131 led to two main conclusions. Firs^{i} the O(N) algorithms are strictly sequential, that is, parallelism n their computation is bounded. More precisely, the main bottleneck in parallel computation of O(N) algorithms is in parallelization of the nonlinear recurrences for computation of the articulated-body inertia, As shown in [12,13], this nonlinear recurrence belongs to a class of recurrences which are well known to be strictly sequential (The reader is referred to [12-14] for a more detailed discussion.) In fact, not surprisingly, various approaches proposed for parallel computation of forward dynamics of open-chain systems are based on the parallelization of the $O(N^3)$ and $O(N^2)$ algorithms [XX, 12,13,19-22) and there seems to be no report on parallel computation of the O(N) algorithms.

The second conclusion in [12] was regarding the existence of a both time- and processor-optimal parallel algorithm, i.e. , an $O(\mathrm{Log}\ N)$ parallel algorithm with O(N) processors, for the problem. If there indeed can be such an Optimal parallel algorithm, then it must be derivable from an O(N) serial algorithm. Since existing O(N) algorithms are strictly sequential, the first step in deriving the optimal parallel algorithm is to develop a new serial O(N) algorithm with efficiency for parallelization in mind, Such an O(N) algorithm can only be developed by a global reformulation of the problem and not an algebraic transformation in the computation of existing O(N) algorithms.

From a physical viewpoint, a given algorithm for the problem can be classified according to its interbody force decomposition strategy, From the standpoint of computation, the algorithm can be classified based on the resulting factorization of the mass matrix which correspond to the specific force decomposition strategy (see [13] for a more detailed discussion,) A new algorithm based on a global reformulation of the problem is then the one that starts with a different and new force decomposition strategy and results in a factorization of mass matrix (see also the brief discussion in § II.C.)

Interestingly, a recently developed iterative algorithm in [23,241 for open-chain system represents such a global reformulation of the problem. It differs from the existing O(N) algorithms in the sense that it is based on a different strategy for force decomposition, In [13,14,25], we have shown that this strategy leads to a new and completely different factorization of M-* in form of Schur Complement, This factorization, in turn, results in a new O(N) algorithm for the problem which is strictly efficient for parallel computation, that is, it is less efficient than other O(N) algorithms for serial computation but, it can be parallelized to achieve the time lower of O(Log N) with O(N) processors.

In this paper, we show" that this factorization of M^{-1} can be used to derive a new Schur Complement factorization of Λ^{-1}. We also show that the application of Schur Complement factorization of M-l and Λ^{-1} for the forward dynamics solution of the single closed-chain systems results in a new O(N) for the problem. This O(N) algorithms, though not competitive for serial computation, can be parallelized, leading to an O(Log N) parallel algorithm with O(N) processors for the problem. To our knowledge, this represents the first optimal parallel algorithm for the problem.

This paper is organized as follows. In §II notation and some preliminaries are presented. The solution procedure for the forward dynamics of a single closed-chain system is reviewed in §III. The Schur Complement factorization of M-* and Λ^{-1} are derived in §IV. Efficient serial implementation of the resulting O(N) algorithm is discussed in §V. Parallel computation of this new O(N) is discussed in §VI wherein a multilevel parallel O(Log N) is also presented, Finally, some concluding remarks are made in §VII.

II Notation and Preliminaries

A. Spatial and Global Notation

In the following derivation, we use spatial notation which, although is slightly different from those in [3,4,17,18,231, allows a clear understanding and comparison of the various algorithms, For the sake of clarity, the spatial quantities are shown with upper-case *italic* letters. Here, only joints with one revolute DOF are considered. However, all the results can be extended to the systems with joints having different and/or more DOFs.

With any vector V, a tensor $\tilde{V}$ can be associated whose representation in any frame is a skew symmetric matrix:

$$\tilde{V} = \begin{bmatrix} 0 & -V_{(z)} & V_{(y)} \\ V_{(z)} & 0 & -V_{(x)} \\ -V_{(y)} & V_{(x)} & 0 \end{bmatrix}$$

where $V_{(x)}$, $V_{(y)}$, and $V_{(z)}$ are the components of V in the frame considered. The tensor $\tilde{V}$ has the properties that $\tilde{V}^T = -\tilde{V}$ and $\tilde{V}_1 V_2 = V_1 x V_2$, i.e., it is a vector cross-product operator (T denotes the transpose). A matrix $\hat{V}$ associated to the vector V is defined as

$$\hat{V} = \begin{bmatrix} U & \tilde{V} \\ 0 & U \end{bmatrix} \text{ and } \hat{V}^T = \begin{bmatrix} U & 0 \\ -\tilde{V} & U \end{bmatrix} \varepsilon \mathbb{R}^{6x6}$$

where here (and through the rest of the paper) U and O stand for unit and zero matrices of appropriate size. The spatial velocities of two rigidly connected points A and B are related as

$$V_A = \hat{P}^T_{A,B} V_B$$

where $P_{A,B}$ denotes the position vector from B to A. The matrix $\hat{P}_{A,B}$ has the properties as

$$\hat{P}_{A,B}\hat{P}_{B,C} = \hat{P}_{A,C} \text{ and } (\hat{P}_{A,B})^{-1} = \hat{P}_{B,A} \tag{1}$$

The spatial forces acting at two rigidly connected points A and B are related:

$$F_B = \hat{P}_{B,A} F_A$$

If the linear and angular velocities of point A are zero then

$$\dot{V}_A = \hat{P}^T_{A,B}\dot{V}_B$$

In general, the spatial inertia of link i about point j is denoted by $I_{i,j}$. The spatial inertia of link i about its center of mass, $I_{i,Ci}$, is given by

$$I_{i,Ci} = \begin{bmatrix} J_i & 0 \\ 0 & m_i U \end{bmatrix} \varepsilon \mathbb{R}^{6\times 6}$$

The spatial inertia of body i about point O_i (designated as I_i) is obtained as

$$I_i = \hat{S}_i I_{i,Ci} \hat{S}^T_i = \begin{bmatrix} U & \tilde{S}_i \\ 0 & U \end{bmatrix} \begin{bmatrix} J_i & 0 \\ 0 & m_i U \end{bmatrix} \begin{bmatrix} U & 0 \\ -\tilde{S}_i & U \end{bmatrix} = \begin{bmatrix} k_i & \tilde{h}_i \\ \tilde{h}^T_i & m_i U \end{bmatrix} \tag{2}$$

which represents the *parallel axis theorem* for propagation of spatial inertia.

In our derivations, we also make use of global matrices and vectors which lead to a compact representation of various factorization. For the sake of clarity, the global quantities are shown with upper-case *script* letters. A bidiagonal block matrix $\mathcal{P}$ is defined as

$$\mathcal{P} = \begin{bmatrix} U & & & & \\ -\hat{P}_{N-1} & U & & 0 & \\ 0 & -\hat{P}_{N-2} & U & & \\ 0 & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & & -\hat{P}_1 & U \end{bmatrix} \varepsilon \mathbb{R}^{6N\times 6N}$$

Note that, according to our notation, $P_{i+1,i} = P_i$. The inverse of $\mathcal{P}$ is a lower triangular block matrix given by

$$\mathcal{P}^{-1} = \begin{bmatrix} U & & & & \\ \hat{P}_{N,N-1} & U & & 0 & \\ \hat{P}_{N,N-2} & \hat{P}_{N-1,N-2} & U & & \\ & & & & \\ \hat{P}_{N,1} & \hat{P}_{N-1,1} & & \hat{P}_{2,1} & U \end{bmatrix} \varepsilon \mathbb{R}^{6N\times 6N}$$

B. An Operator Expression of Jacobian Matrix

Following the treatment in [10], a factorization of Jacobian matrix by using our notation can be derived as follows. The velocity propagation for a serial chain of interconnected rigid body is given by (Fig. 1)

$$V_i - \hat{P}^T_{i-1,i} V_{i-1} = H_i Q_i \tag{3}$$

which, by using the matrix $\mathcal{P}$, can be expressed in a global form as

$$\mathcal{P}^T V = \mathcal{H}Q \Rightarrow V = (\mathcal{P}^T)^{-1}\mathcal{H}Q \tag{4}$$

The EE spatial velocity, V_{N+1}, is obtained by writing Eq. (3) for i = N+1 as

$$V_{N+1} - \hat{P}^T_N V_N = 0 \Rightarrow V_{N+1} = \hat{P}^T_N V_N \tag{5}$$

Let us define a matrix $\beta = [\hat{P}^T_N \; 0 \; 0 \; . \; . \; . \; 0] \epsilon \mathbb{R}^{6x6N}$. From Eqs. (4)-(5), we get

$$V_{N+1} = \beta V \beta (\mathcal{P}^T)^{-1}\mathcal{H}Q \tag{6}$$

The Jacobian matrix is defined by relating the EE spatial velocity and joint velocities as

$$V_{N+1} \cdot \mathcal{J}Q \tag{7}$$

From Eqs. (6)-(7) a factorization of Jacobian matrix is then derived as

$$\mathcal{J} = \beta(\mathcal{P}^T)^{-1}\mathcal{H} \tag{8}$$

C. Equations of Motion

The equations of motion for a single (open- or closed-chain) arm are given by

$$\mathcal{M}\dot{Q} = \mathcal{T} - b(\theta, Q) - \mathcal{J}^t F_{N+1} \tag{9}$$

Defining

$$b'(\theta, Q, F_{N+1}) = b(\theta, Q \quad + \mathcal{J}^T F_{N+1}, \text{ and}$$

$$\mathcal{F}_T = \mathcal{T} - b' \; (9, Q, F_{N+1}$$

Eq. () can be written as

$$\mathcal{M}\dot{Q} = \mathcal{F}_T \Rightarrow \quad Q \quad = \quad M\text{-}\%* \tag{lo}$$

The vector $b'(\theta, Q, F_{N+1})$ represents the contribution of nonlinear terms and the external spatial force (F_{N+1}). For an open-chain arm, F_{N+1} is specified and hence the vector $b'(\theta, Q, F_{N+1})$ can be computed by using the Newton-Euler (N-E) algorithm [26] while setting Q to zero. In Eq. (10), $\mathcal{F}_T \triangleq col\{F_{Ti}\}\epsilon\mathbb{R}^{Nx1}$ represents the acceleration-dependent component of the control force.

In deriving the factorization of mass matrix, it is assumed that the vector

$b'(\theta, Q, F_{N+1})$ and subsequently $\mathcal{F}_T$ are explicitly computed. Therefore, the multibody system can be assumed as a system at rest which upon the application of the control force $\mathcal{F}_T$ accelerates in space, The propagation of accelerations and forces among the links of serial chain are then given by

$$\dot{V}_i = \hat{P}^T_{i,i}\dot{V}_{i-1} + H_i\dot{Q}_i \tag{11}$$

$$F_i = I_i\dot{V}_i + \hat{P}_i F_{i+1} \tag{12}$$

Equations (11)-(12) represent the simplified N-E algorithm (with nonlinear terms being excluded) for the serial chain. Note that, the force decomposition strategy of the N-E algorithm is given by Eq. (12). As shown in [17,181, this strategy leads to a specific factorization of $\mathcal{M}$. It should be pointed out that the recursive factorization of A-' in [16-18] can be shown [131 to be a factorization resulting from the specific force decomposition strategy of the Articulated-Body Inertia algorithm given by Eq. (26) in [15].

III. Dynamics of a Robot Arm in Contact with Environment,

A. Dynamic Equations of Motion

Our problem statement of the dynamics of a single robot arm in contact with the environment mainly follows the treatment presented in [3,6,7]. For closed-chain systems Eq. (9) is written as

$$\mathcal{M}\dot{Q} = \mathcal{F}'_T - \mathcal{J}^T F_{N+1} \tag{13}$$

where F_{N+1} is the unknown spatial contact force exerted by the EE and

$$\mathcal{F}'_T = \mathcal{T} - b(\theta, Q)$$

The vector $b(\theta, Q)$ can be computed by using the N-E algorithm while setting both Q and F_{N+1} to zero. Equation (13) can be rewritten as

$$\dot{Q} = \mathcal{M}^{-1}\mathcal{F}'_T - \mathcal{M}^{-1}\mathcal{J}^T F_{N+1} = \dot{Q}_O - \dot{Q}_C \tag{14}$$

$$\dot{Q}_O = \mathcal{M}^{-1}\mathcal{F}'_T \tag{15}$$

$$\dot{Q}_C = \mathcal{M}^{-1}\mathcal{J}^T F_{N+1} \tag{16}$$

where Q_O is the vector of joint accelerat ons of unconstrained, or open-chain, system and $\dot{Q}_C$ is the vector of joint acce erations resulting from the spatial contact force. From Eq. (7 the spatial acceleration of EE is derived as

$$V_{N+1} = \mathcal{J}\dot{Q} + \dot{\mathcal{J}}Q \tag{17}$$

By substituting Eqs. (14)-(16) into 17) it follows that

$$\dot{V}_{N+1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{F}_T + \dot{\mathcal{J}}Q - \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^T F_{N+1} = \dot{V}_{ON+} - V_{CN+1} \tag{18}$$

$$\dot{V}_{ON+1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{F}_T + \dot{\mathcal{J}}Q \tag{19}$$

$$\dot{V}_{CN+1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^T F_{N+1} \tag{20}$$

where V_{ON+1} is the EE spatial acceleration of open-chain arm and $\dot{V}_{CN+1}$ is the EE spatial acceleration resulting from the contact force. The matrix

$$\Lambda^{-1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^T \varepsilon \mathbb{R}^{6\times6} \tag{21}$$

is the inverse of operational space mass matrix [1], The conditions for positive definiteness of matrix Λ^{-1} is discussed in the appendix, Throughout the rest of the paper it is assumed that Λ^{-1} is positive definite.

B. Model of Contact

The EE of an open-chain arm has six DOFs. The contact with the environment constrains the motion of the EE and results in the loss of DOFs. Modeling of contact has been discussed in literature (see, for example, [XX.XX]). Following [3,6,7] the contact can be modeled as a multiple-DOFs joint (joint N+1) connecting the EE and the environment. Let H_{N+1} and W_{N+1} stand for the map matrices of joint N+1 representing the motion (unconstrained) and constraint vector subspaces, respectively. The two vector spaces are orthogonal, that is,

$$H^T_{N+1} W_{N+1} = 0 \text{ and } W^T_{N+1} H_{N+1} = O \quad : \tag{22}$$

For the **sake** of simplicity and with no loss of generality, let us assume, as in [6,7 , that H_{N+1} and W_{N+1} are orthonormal, that is,

$$H^T_{N+1} H_{N+1} = U \text{ and } W^T_{N+1} W_{N+1} = U$$

Let nf and nc denote the number of DOFS and the number of degrees of constraint (DOC) of the EE, respectively, with nf + nc = 6. In this case, $H_{N+1} \varepsilon \mathbb{R}^{6\times nf}$ and $W_{N+1} \varepsilon \mathbb{R}^{6\times nc}$. Note that, as will be seen in §IV, this modeling of

Joint N+1 is similar to the way that other joints of arm are modeled,

The EE spatial acceleration and force are given by [6,7]

$$\dot{V}_{N+1} = H_{N+1}G_F + W_{N+1}G_C \tag{23}$$

$$F_{N+1} = H_{N+1}K_F + W_{N+1}K_C \tag{24}$$

where G_F and $K_F \varepsilon \mathbb{R}^{nf}$ and G_C and $K_C \varepsilon \mathbb{R}^{nc}$. Following [6,7], two types of contacts are considered:

Type I. G_C and K_F are specified and G_F and K_C need to be computed,

Type II. G_C is specified and

$$K_F = \phi K_C + d \tag{25}$$

where matrix ϕ and vector d are known. For this type, G_F, K_F, and K_C need to be computed.

C. Forward Dynamics Solution

We briefly review the dynamic solution for the two types of contact as presented in [6,7]. Our main purpose, however, is to analyze the computational steps involved in the solution which is essential for developing serial and parallel algorithms discussed in §IV and §V.

From Eqs. (18)-(21) it follows that

$$V_{0N+1} - \Lambda^{-1}F_{N+1} = \dot{V}_{N+1} \tag{26}$$

For Type I contact, substituting Eqs. (23) and (24) into Eq. (26) leads to

$$\dot{V}_{0N+1} - \Lambda^{-1}F_{N+1} = H_{N+1}G_F + W_{N+1}G_C \tag{27}$$

Multiplying both sides of Eq. (27) by W^T_{N+1} and using Eq. (22) gives:

$$W^T_{N+1}\dot{V}_{N+1} - W^T_{N+1}\Lambda^{-1}F_{N+1} = G_C \tag{28}$$

from which, after some manipulation, it follows that

$$(W^T_{N+1}\Lambda^{-1}W_{N+1})K_C = W^T_{N+1}\dot{V}_{N+1} - W^T_{N+1}\Lambda^{-1}H_{N+1}K_F - G_C \tag{29}$$

If Λ^{-1} is positive definite then the matrix $W^T_{N+1}\Lambda^{-1}W_{N+1}\varepsilon\mathbb{R}^{nc\times nc}$ *is* also positive definite (see Appendix) and hence Eq. (29) can be solved for K_C. Note that, if the right-hand side of Eq. (29) is obtained then the computation of K_C requires the solution of an nc×nc SPD linear system.

For Type 11 contact, by substituting Eqs. (23)-(25) into Eq. (26), it can be shown that [6]

$$((W^T_{N+1}\Lambda^{-1})(H_{N+1}\phi + W_{N+1}))K_C = W^T_{N+1}V_{N+1} - W^T_{N+1}\Lambda^{-1}H_{N+1}d - G_C$$

which, assuming that the coefficient matrix is positive definite, can be solved for K_C.K_F and F_{N+1} can then be computed from Eqs. (25) and (24).

As can be seen, the explicit computation of the vector $\dot{\mathcal{J}}Q$ is needed in the solution procedure. It seems, however, that less attention has been paid in the literature to the efficient computation of this vector. In [27] a method for computation of the matrix $\dot{\mathcal{J}}$ is proposed. Note, however, that the explicit computation of $\dot{\mathcal{J}}$ is not needed. In fact, as noted by Featherstone [15] and discussed in [28], the vector $\dot{\mathcal{J}}Q$ can be obtained with" almost no additional cost as a byproduct of the computation of the vector b′ (θ,Q). To see this, note that in Eq. (17) if Q is set to zero then the vector $\dot{\mathcal{J}}Q$ represents the spatial acceleration of the EE resulting from nonzero vector of joint velocities (Q). Recall that the vector b'(θ,Q) is obtained by computing the N-E algorithm while setting the vector $\dot{Q}$ to zero, Therefore, if the N-E algorithm is slightly modified so that the spatial acceleration of the EE, denoted by $\dot{V}'_{0N+1}$, is also computed then we simply have $\dot{V}'_{0N+1} = \dot{\mathcal{J}}Q$.

Based on the above discussion, the computational steps of the dynamic solution procedure are summarized in Table I

D. Extension to other Models of Contact

We considered the specific model of contact in [6,7] since it represents a good example which can be used to highlight the computational requirements of the forward dynamics solution of the single closed-chain system. Other models of contact will lead to a rather slight modification in the computation. To see this, note that, a different model of contact results in a different strategy for computation of the contact force, F_{N+1}. Once this force is obtained the rest of the computations remain the same as in Table I. In

Step I: Compute $\mathcal{F}'_T$

1. Compute $b'(\theta, Q)$ and $\dot{V}'_{ON+1} = \dot{\mathcal{J}}Q$ by using the N-E algorithm while setting Q and F_{N+1} to zero.

2. Compute $\mathcal{F}'_T = \mathcal{T} - b'(\theta, Q)$ (30)

Step II: Compute $\dot{V}_{ON+1}$

1. Compute $\dot{Q}_O = \mathcal{M}^{-1}\mathcal{F}'_T$ (31)

2. Compute $\dot{V}''_{ON+1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{F}'_T = \mathcal{J}\dot{Q}_O$ (32)

3. Compute $\dot{V}_{ON+1} = \dot{V}''_{ON+1} + \dot{V}'_{ON+1}$ (33)

Step III: Compute Λ^{-1}

Step IV: Compute F_{N+1}

1. Solve Eq. (29) for $K_{C''}$

2. Compute F_{N+1} from Eq. (24).

Step V: Compute Q

1. Compute $\mathcal{T}' = \mathcal{J}^T F_{N+1}$ (34)

2. Compute $\dot{Q}_C = \mathcal{M}^{-1}\mathcal{J}^T F_{N+1} = M\text{-}19'$. *(35)*

3. Compute $\dot{Q} = \dot{Q}_O - \dot{Q}_C$. (36)

Table I. The Computational Steps of Forward Dynamics Solution Procedure

general, Eq. (26) represents six equations in twelve unknowns, i.e., the twelve components of F_{N+1} and $\dot{V}_{N+1}$. Note that, the computation of $\dot{V}_{ON+1}$ is independent of the model of contact considered and only requires the solution of the open-chain system. If the problem is well posed then another set of six equations can be derived leading to the complete solution for F_{N+1} and $\dot{V}_{N+1}$. In the following a different model of contact is briefly discussed.

Lathrop [51 considered several examples of contact and proposed a more general model in which F_{N+1} and $\dot{V}_{N+1}$ are given (in some frame) by

$$\dot{V}_{N+1} = M_1\Omega + \kappa_1 \quad (37)$$

$$F_{N+1} = M_2\Omega + \kappa_2 \quad (38)$$

where the matrices M_1 and $M_2 \epsilon \mathbb{R}^{6\times6}$ and the vectors κ and $\kappa_2 \epsilon \mathbb{R}^6$ are known. The unknown vector $\Omega \epsilon \mathbb{R}^6$ represents the six independent motion and force DOFs. Clearly, once Ω is obtained F_{N+1} and $\dot{V}_{N+1}$ canbecomputed from Eqs. (37)-(381 and the problem is then reduced to that in previous section, By replacing Eqs. (37)-(38) into Eq. (26) it follows that

$$(\Lambda^{-1}M_2 + M_1)\Omega = \dot{V}_{ON+1} - (\kappa_1 + \Lambda^{-1}\kappa_2) \tag{39}$$

If the problem is well posed, i.e., the coefficient matrix is positive definite, then Ω can be obtained by solving the 6x6 linear system in Eq. (39).

IV. Schur Complement Factorization of *A-l* and Λ^{-1}

A. The Interbody Force Decomposition Strategy

The iterative algorithms in [23,24] for forward dynamics solution of open-chain arms are based on a decomposition of interbody force of the form:

$$F_i = H_i F_{Ti} + W_i F_{Si} \quad (401$$

where F_{Si} is the constraint force and *Vi* is the orthogonal complement of H_i [29,301, that is,

$$W_i^T H_i = O \text{ and } H_i^T W_i = O \tag{41}$$

For a joint i with multiple DOFs, say ni<6 DOFs,$H_i \epsilon \mathbb{R}^{6\times ni}$ and $W_i \epsilon \mathbb{R}^{6\times(6-ni)}$. Insofar as the axes of DOFS are orthogonal (which is the case considered in this paper) the matrix H_i *is* a projection matrix [29] and hence

$$H_i^T H_i = U \tag{42}$$

It then follows that the matrix W_i *is* also a projection matrix [29,301, i.e.,

$$W_i^T W_i = u \tag{43}$$

$$H_i H_i^T + W_i W_i^T = U \tag{44}$$

An example of matrices H_i and W, for one DOF revolute joints is given in §V.A. For a more detailed discussion on these matrices see [29,301.

The decomposition in Eq. (40) seems to be more physically intuitive than that of the Articulated-Body Inertia algorithm, given by Eq. (40) in [15], since it expresses the interbody force in terms of two physical components:

the control (or, working) force and the constraint (or, nonworking) force. That such force decomposition has not been considered as a viable alternative for deriving algorithms for *direct* serial and parallel solution of the problem is not surprising. The decomposition in Eq. (40) naturally leads to explicit computation of the constraint (and interbody) forces. In fact, researchers have often argued that since the constraint forces are nonworking forces their explicit evaluation, which leads to the computational inefficiency, should be avoided. While this argument is in general valid for serial computation- which is also supported by the results in [13,141 for open-chain systems and the results of this paper for single closed-chain systems- the decomposition in Eq. (40) leads to new factorization of $\mathcal{M}^{-1}$ and Λ^{-1} and subsequently optimal parallel algorithms for forward dynamics of both open- and closed-chain arms.

B. A Schur Complement Factorization of Λ^{-1}

In [13,14], we have shown that the force decomposition in Eq. (40) leads to a new factorization of $\mathcal{M}^{-1}$ and subsequently a new $O(N)$ algorithm for the forward dynamics of open-chain arms. We briefly review this factorization of $\mathcal{M}^{-1}$ since not only it is needed for solution of closed-chain arms but, more importantly, it is also essential in deriving the factorization of Λ^{-1}.

To begin, let us define following global matrix and vector for i = N to 1:
$\mathcal{W} \overset{\Delta}{=} \text{diag}\{W_i\} \epsilon \mathbb{R}^{6N\times 5N}$ and $\mathcal{F}_S \overset{\Delta}{=} \text{col}\{F_{Si}\} \epsilon \mathbb{R}^{5N}$

Equations (11)-(12) and (40)-(44) can be now written in global form as

$$\mathcal{P}^T\dot{\mathcal{V}} = \mathcal{H}\dot{Q} \tag{45}$$

$$\mathcal{P}\mathcal{F} = \mathcal{I}\dot{\mathcal{V}} \tag{46}$$

$$\mathcal{F} = \mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S \tag{47}$$

$$\mathcal{W}^T\mathcal{H} = 0 \text{ and } \mathcal{H}^T\mathcal{W} = 0 \tag{48}$$

$$\mathcal{H}^T\mathcal{H} = U \text{ and } W^TW = U \tag{49}$$

$$\mathcal{H}\mathcal{H}^T + \mathcal{W}\mathcal{W}^T = U \tag{50}$$

From Eqs. (45), (46), and (48) it follows that

$$\dot{\mathcal{V}} = \mathcal{G}^{-1}\mathcal{P}\mathcal{F} \quad (51)$$

$$\mathcal{W}^T\mathcal{P}^T\dot{\mathcal{V}} = \mathcal{W}^T\mathcal{H}\dot{Q} = 0 \quad (52)$$

Replacing Eq. (51, into Eq. (52), we get

$$\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{F} = 0 \quad (53)$$

Substituting Eq. (47) into Eq. (53) yields

$$\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}(\mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S) = O \Rightarrow \mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S = -\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T, \text{ or} \quad (54)$$

$$\mathcal{A}\mathcal{F}_S = -\mathcal{B}\mathcal{F}_T \quad (55)$$

where $\mathcal{A} \triangleq \mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W} \in \mathbb{R}^{5N\times5N}$ and $\mathcal{B} \triangleq \mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{5N\times N}$ are block tridiagonal matrices. From Eqs. *(55)* and *(47)* it follows that

$$3 = \left(R - \mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \quad (56)$$

andsubstituting Eq. *(56)* into Eq. (51) leads to

$$\dot{\mathcal{V}} = 9^{-1}3\left(\mathcal{H} - \mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \quad (57)$$

By multiplying both sides of Eq. (45) by $\mathcal{H}$ and using Eq. (49) Q is computed as

$$\mathcal{H}^T\mathcal{H}\dot{Q} = \mathcal{H}^T\mathcal{P}^T\dot{\mathcal{V}} \Rightarrow Q = \mathcal{H}^T\mathcal{P}^T\dot{\mathcal{V}} \quad (58)$$

Finally, by replacing Eq. (57) into (58) it follows that

$$Q = \left[\mathcal{H}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H}\right]\mathcal{F}_T \quad (59)$$

In comparison with Eq. (10), an operator factorization of *M-l,* in terms of its decomposition into a set of simpler operators, is given by

$$M^{-1} = \mathcal{H}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H} \quad (60)$$

Let $\mathcal{C} \triangleq \mathcal{H}^T\mathcal{P}^T\mathcal{G}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{N\times N}$ $\mathcal{M}^{-1}$ can now be expressed as

$$M^{-1} = \mathcal{C} - \mathcal{B}^T\mathcal{A}^{-1}\mathcal{B} \quad (61)$$

The matrix $\mathcal{C}$, similar to $\mathcal{A}$ and $\mathcal{B}$ is block tridiagonal. Furthermore, as shown in the appendix, $\mathcal{A}$ and $\mathcal{C}$ are symmetric and positive definite (SPD). This guarantees the existence of $\mathcal{A}^{-1}$

The operator form of $\mathcal{M}^{-1}$ given by Eq. (61) represents an interesting mathematical construct. To see this, note that, if a matrix $\mathcal{L}$ is defined as

$$\mathcal{L} \triangleq \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{B}^T & \mathcal{C} \end{bmatrix} \in \mathbb{R}^{6N\times6N}$$

then $\mathcal{C} - \mathcal{B}^T\mathcal{A}^{-1}\mathcal{B}$ is the *Schur Complement* of $\mathcal{A}$ in $\mathcal{L}$ [311. The structure of matrix $\mathcal{L}$ not only provides a deeper physical insight into the computation but it also motivates a different and a much simpler approach for derivation of the factorization of *M-l* and its associated O(N) algorithm [13,321.

It is worth pointing out that by using the matrix identity

$$(E - XDY)^{-1} = E^{-1} + E^{-1}X(D^{-1} - YE^{-1}X)^{-1}YE^{-1} \tag{62}$$

in Eqs. (60)-(61), an operator expression of *M* can be obtained as

$$\begin{aligned}\mathcal{M} &= \mathcal{C}^{-1} + \mathcal{C}^{-1}\mathcal{B}^T(\mathcal{A}^{-1} - \mathcal{B}\mathcal{C}^{-1}\mathcal{B}^T)\mathcal{B}\mathcal{C}^{-1} \\ &= (\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})^{-1} + (\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W})\Big((\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W}) - (\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H}) \\ &\quad (\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W})\Big)^{-1}(\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})(\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})^{-1}\end{aligned} \tag{63}$$

However, this operator expression of ***M*** is significantly more complex and its associated algorithm is less efficient (in terms of number of operat. ens) than other operator expressions and their associated algorithms given in 17,18]. This clearly indicates that the force decomposition given by Eq. (42) leads more naturally and efficiently to the computation of *M-l* rather than *M.*

C. A Schur Complement Factorization of Λ^{-1}

The new factorization of ***M-l*** directly results in a new factorization of Λ^{-1}. This factorization is derived by substituting the factorization of $\mathcal{J}$, given by Eq. (8), and ***M-l,*** given by Eqs. (60)-(61), into Eq. (21):

$$\Lambda^{-1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^T = \beta(\mathcal{P}^T)^{-1}\mathcal{H}(\mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{H})\mathcal{H}^T\mathcal{P}^{-1}\beta^T$$

which can be written as

$$\Lambda^{-1} = \beta((\mathcal{P}^T)^{-1}(\mathcal{H}\mathcal{H}^T)\mathcal{P}^T(\mathcal{J}^{-1} - \mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1})\mathcal{P}(\mathcal{H}\mathcal{H}^T)\mathcal{P}^{-1})\beta^T \tag{64}$$

The key to simplification of this expression is the fact that, from Eq. (50), we have

$$\mathcal{H}\mathcal{H}^T = \mathcal{U} - \mathcal{W}\mathcal{W}^T \tag{65}$$

By replacing Eq. (65) into Eq. (64) and after some involved algebraic manipulations, a simple operator expression of Λ^{-1} is derived as

$$\Lambda^{-1} = \beta\mathcal{J}^{-1}\beta^T - \beta\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^T\mathcal{P}^T\mathcal{J}^{-1}\beta^T \tag{66}$$

This expression can be further simplified since

$$\mathcal{E}^T = \beta\mathcal{I}^{-1}\mathcal{P}W = [\hat{P}^T_N I^{-1}_N W_N \; 0 \; 0 \; . \; . \; . \; 0] \epsilon \mathbb{R}^{6\times 5N} \tag{67}$$

$$\mathcal{D} = \beta\mathcal{I}^{-1}\beta^T = \hat{P}^T_N I^{-1}_N \hat{P}_N \tag{68}$$

The parallel axis theorem in Eq. (2) can be also used for propagation of the inverse of spatial inertias. To this end, by using Eq, (l), Eq. (68) can be rewritten as

$$\mathcal{D} = ((\hat{P}_N)^{-1}(I_N)(\hat{P}^T_N)^{-1})^{-1} = (\hat{P}_{N+1,N} I_N \hat{P}^T_{N+1,N})^{-1} = I^{-1}_{N,N+1}$$

which implies that the matrix $\mathcal{D}$ is just the inverse of spatial inertia of link N about point O_{N+1}.

This factorization of Λ^{-1} can be written in form of Schur Complement as

$$A^{-1} = \mathcal{D} - \mathcal{E}^T\mathcal{A}^{-1}\mathcal{E} \tag{70}$$

Note that the matrix $\mathcal{A}$ is the same as in Eq. (61). Let us define a matrix $\mathcal{L}'$:

$$\mathcal{L}' = \begin{bmatrix} \mathcal{A} & -\mathcal{E} \\ \mathcal{E}^T & \mathcal{D} \end{bmatrix} \epsilon \mathbb{R}^{6n\times 6n}$$

A^{-1} is then the Schur Complement of $\mathcal{A}$ in $\mathcal{L}'$.

The factorization of A-* has a structure similar to that of $\mathcal{M}^{-1}$sinceitis also described in form of Schur Complement. As for M^{-1}, the structure of matrix $\mathcal{L}'$ not only enables a simple physical interpretation of this factorization but also motivates an alternate and somehow simpler approach for its derivation [311. However, it should be emphasized that the similarity in the factorization of *M-l* and Λ^{-1} is not limited to their analytical form (i.e., the Schur Complement form) but it further extends to their physical interpretation. To see this, let us rewrite *M-l* and A-l as

$$\mathcal{M}^{-1} = \mathcal{H}^T\mathcal{P}^T(\mathcal{I}^{-1} - \mathcal{I}^{-1}\mathcal{P}W(W^T\mathcal{P}^T\mathcal{I}^{-1}\mathcal{P}W)^{-1}W^T\mathcal{P}^T\mathcal{I}^{-1})\mathcal{P}\mathcal{H}$$

$$A^{-1} = \beta(\mathcal{I}^{-1} - \mathcal{I}^{-1}\mathcal{P}W(W^T\mathcal{P}^T\mathcal{I}^{-1}\mathcal{P}W)^{-1}W^T\mathcal{P}^T\mathcal{I}^{-1})\beta^T$$

Let us also define a matrix $\mathcal{K}$ as

$$\mathcal{K} = \mathcal{I}^{-1} - \mathcal{I}^{-1}\mathcal{P}W(W^T\mathcal{P}^T\mathcal{I}^{-1}\mathcal{P}W)^{-1}W^T\mathcal{P}^T\mathcal{I}^{-1}$$

$\boldsymbol{M}^{-1}$ and A^{-1} can now be expressed as

$$\mathcal{M}^{-1} = \mathcal{H}^T\mathcal{P}^T\mathcal{K}\mathcal{P}\mathcal{H}$$

$$\Lambda^{-1} = \beta\mathcal{K}\beta^T$$

As shown in [31], the matrix $\mathcal{K}$ has a simple physical interpretation. The fact that $\mathcal{M}^{-1}$ and Λ^{-1} can be both derived from $\mathcal{K}$ then allows a unified and alternate physical interpretation of factorization of $\mathcal{M}^{-1}$ and A^{-1} based on the physical interpretation of matrix $\mathcal{K}$.

From a computational perspective, a main advantage of this structural similarity resides in the improved efficiency in both serial and parallel computation, As shown in §V and §VI, for the cases (such as the one in this paper) wherein the computation of both $\mathcal{M}^{-1}$ and Λ^{-1} is needed, this structural similarity can be exploited to increase the computational efficiency

V. A Serial O(N) Algorithm for Forward Dynamics of Single Closed-Chain Arm

In this section, a serial algorithm for the problem based on the Schur Complement factorization of ***M-I*** and Λ^{-1} is discussed. This discussion provides a deeper insight into the structure of computation while using these factorization. It also highlights some issues that need to be considered to achieve a greater efficiency in both serial and parallel computations.

A. Operator Application of *M-I*

As shown in [25], the explicit computation of ***M-'*** can be performed in $O(N^2)$ steps. Once $\mathcal{M}^{-1}$ is computed, its multiplication by a vector also requires $O(N^2)$ operations. Note, however, that only the result of the multiplication of M^{-1} by a vector, as in Eqs. (31) and (35), which corresponds to an operator application of $\mathcal{M}^{-1}$, rather than its explicit computation is needed. This operator application, as shown below, can be performed more efficiently in only O(N) steps.

For the sake of generality, we consider an application of $\mathcal{M}^{-1}$ of the form:

$$\dot{Q}_G = \mathcal{M}^{-1}\mathcal{F}_G \qquad (71)$$

For both serial and parallel computation it is more efficient to rewrite the above equation, after substituting Eq. (60), as

$$\dot{Q}_G = \mathcal{H}^T\mathcal{P}^T \left\{ u - \mathcal{J}^{-1}\mathcal{P}W(W^T\mathcal{P}^T\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^T\mathcal{P}^T \right\}_1 \mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_G \qquad (72)$$

Here, the key to achieving a greater computational efficiency is to perform matrix-vector multiplication instead of matrix-matrix multiplication. In this regard, the products of matrices in Eq. (72) do not need to be computed explicitly and only the explicit computation of matrix $\mathcal{A}$ is needed. Given $\mathcal{F}_G$, the computational steps in implementing Eq. (72) then consist of a sequence of matrix-vector multiplications and a vector addition wherein the matrices, except for $\mathcal{A}^{-1}$, are either diagonal or bidiagonal. Multiplication of a vector by matrix $\mathcal{A}^{-1}$ is equivalent to the solution of a SPD block tridiagonal system.

Thus far, the factorization of $\mathcal{M}^{-1}$ has been presented in a coordinate-free form, Before its implementation, however, the tensors and vectors involved in its computation should be projected onto a suitable frame. The choice of the appropriate frame and the way that the projection is performed significantly affect the efficiency of both serial and parallel computations.

If the rotation of the one-DOF revolute joint i is given as the one about the Z axis of frame i then

$$H_i = \begin{bmatrix} Z_i \\ 0 \end{bmatrix} \varepsilon \mathbb{R}^6$$

The matrices H_i and W_i in frame i are given as

$${}^iH_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } {}^iW_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplication of any vector or matrix by ${}^iH_i$ and ${}^iW_i$ does not require any computation but an appropriate permutation of the elements of the vector or the matrix. However, in any other frame W_i has a dense structure and its multiplication by another matrix requires a rather significant amount of operations. This clearly indicates that any projection of equations should fully exploit the sparse structure of ${}^{i}H_i$ and particularly of ${}^{i}W_i$.

The matrix $\mathcal{A}$ and its elements are given as

$$\mathcal{A} = \text{Tridiag}\,[B_i,\ A_i, B^T_{i}]$$

$$A_i = W_i^T(I_i^{-1} + \hat{P}^T_{i-1} I^{-1}_{i-1}\hat{P}_{i-1})W_i \qquad i = N, N-1, ,\ldots, 1 \tag{73}$$

$$B_i = -W_i^T I_i^{-1}\hat{P}_i W_{i+1} \qquad i = N-1, N-2, \ldots 1 \tag{74}$$

Generalizing the result of Eq. (69), it follows that

$$\hat{P}^T_{i-1} I^{-1}_{i-1}\hat{P}_{i-1} = I^{-1}_{i-1,i}$$

$$A_i = W_i^T(I_i^{-1} + I^{-1}_{i-1,i})W_i$$

To exploit the structure of W_i, the submatrices A_i are computed in frame i as

$$A_i = {}^{i}W_i^T({}^{i}I_i^{-1} + {}^{i}I^{-1}_{i-1,i}){}^{i}W_i \tag{75}$$

Note that, $I_{i-1,i}$ and its inverse are constant in frame i and hence can be precomputed. From Eq. (2), I_i^{-1} can be computed as

$$I_i = \hat{S}_i I_{i,ci}\hat{S}_i^T \Rightarrow I_i^{-1} = (\hat{S}_i^T)^{-1} I^{-1}_{i,ci}(\hat{S}_i)^{-1} = \begin{bmatrix} J_i^{-1} & J_i^{-1}\tilde{S}_i \\ -\tilde{S}_i J_i^{-1} & -\tilde{S}_i J_i^{-1}\tilde{S}_i + (1/m_i)U \end{bmatrix} \tag{76}$$

Both I_i and I_i^{-1} are constant in frame i+1. However, while I_i has a simple and sparse structure, I_i^{-1} has a dense structure, Rather than projecting ${}^{i+1}I_i^{-1}$ onto frame i, it is more efficient to first project J_i^{-1} and S_i onto frame i as

$${}^{i}J_i = C(i, i+1)\,{}^{i+1}J_i C(i+1, i) \Rightarrow ({}^{i}J_i)^{-1} = C(i, i+1)({}^{i+1}J_i)^{-1}C(i+1, i) \tag{77}$$

$${}^{i}S_i = C(i, i+1)\,{}^{i+1}S_i \tag{78}$$

and then compute $({}^{i}I_i)^{-1}$ in frame i according to Eq. (76). The computation of B_i is also performed in frame i as follows. Let us define

$$\Psi_i \triangleq I_i^{-1}\hat{P}_i W_{i+1} \Rightarrow {}^{i+1}\Psi_i = ({}^{i+1}I_i)^{-1}\,{}^{i+1}\hat{P}_i\,{}^{i+1}W_{i+1}$$

$$\mathcal{C}(i, i+1) = \begin{bmatrix} C(i,i+1) & 0 \\ 0 & C(i,i+1) \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

The matrix Ψ_i is constant in frame i+1 and can be precomputed. It is projected onto frame i as

$${}^{i}\Psi_i = \mathcal{C}(i, i+1)\,{}^{i+1}\Psi_i \tag{79}$$

Then, B_i can be computed as

$$B_i = -{}^{i}W_{i}^{T\,i}\Psi_{i} \tag{80}$$

which does not need any computation but a permutation of matrix ${}^iW_i$. In order to further exploit the structure of ${}^iW_i$ and ${}^iH_i$, the rest of the computations in Eq. (72) is also performed in frame i. To do so, let us define a matrix $\mathcal{R}$

$$\mathcal{R} = \begin{bmatrix} U & & & & \\ -C(N-1,N)\,{}^{N}\hat{P}_{N-1} & U & & 0 & \\ 0 & -C(N-2,N-1)\,{}^{N-1}\hat{P}_{N-2} & U & & \\ 0 & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & & -C(1,2)\,{}^{2}\hat{P}_{1} & U \end{bmatrix} \in \mathbb{R}^{6N\times 6N}$$

Let us also define following projections:

$\mathcal{H} = \mathrm{diag}\{{}^{i}H_{i}\}$; $\mathcal{W} = \mathrm{diag}\{{}^{i}W_{i}\}$; and $\underline{\mathcal{I}}^{-1} = \mathrm{diag}\{{}^{i}I_{i}^{-1}\}$

Through the rest of the paper, an underlined global vector or global block diagonal matrix indicates that the ith element of the vector or the matrix is described in the ith frame. Equation (72) is now rewritten as

$$\dot{Q}_G = \underline{\mathcal{H}}^T\mathcal{R}^T(U - \underline{\mathcal{I}}^{-1}\mathcal{R}\underline{\mathcal{W}}\mathcal{A}^{-1}\underline{\mathcal{W}}^T\mathcal{R}^T)\underline{\mathcal{I}}^{-1}\mathcal{R}\underline{\mathcal{H}}\mathcal{F}_G \tag{81}$$

which implies that all the computations are performed in frame i.

The most computation-intensive part in implementing Eq. (81) is the solution of the SPD block tridiagonal system. The block tridiagonal system can be solved by both Block Cyclic Reduction (BCR) algorithm [33-35] and block LDL^T factorization [361 in O(N) steps. However, for serial computation the latter algorithm is more efficient [34]. Using the block LDL^T algorithm, the solution of the SPD block tridiagonal system consists of three steps [361: factorization, forward elimination, and back substitution, wherein the first step is computationally more expensive than the second and third steps. Note, however, that for the two applications of $\mathcal{M}^{-1}$ in Eqs. (31) and (35), the factorization needs to be performed once.

The projection scheme discussed above and the explicit computation of matrix $\mathcal{A}$ can be performed in O(N) steps. The matrix-vector multiplications in Eq. (81), which involve block diagonal and block bid agonal matrices, as well as the vector addition can be also performed in O(N) steps. Given the O(N) cost of the SPD block tridiagonal system solution, i then follows that the computational complexity of implementing Eq. (81) is of O(N).

B. Computation of Λ^{-1}

Both the explicit computation of Λ^{-1} and its operator applications can be performed in O(N) steps. Here, the explicit computation of Λ^{-1} is considered since its different operators applications in Eq. (29) require simple matrix-vector operations which can be computed with flat costs.

Anticipating the fact that the application of Λ^{-1} is needed in frame N+1 (e.g., for the case considered in §III, W_{N+1}, H_{N+1}, and the Vectors K_C, K_F, etc. , are given in frame N+1), the matrix Λ^{-1} is computed in this frame as

$$\Lambda^{-1} = {}^{N+1}\mathcal{I}^{-1}_{N,N+1} - \mathcal{E}^T\mathcal{A}^{-1}\mathcal{E} \tag{82}$$

wherein the matrix $\mathcal{E}$, or more precisely, the only nonzero element of $\mathcal{E}$, i.e., $\mathcal{E}_N$, is computed as

$$\mathcal{E}_N = {}^{N}W^T_N C(N,N+1)({}^{N+1}\mathcal{I}^{-1}_N {}^{N+1}\hat{P}_N) \tag{83}$$

Note that, with this projection the matrix $\mathcal{E}^T\mathcal{A}^{-1}\mathcal{E}$ and subsequently Λ^{-1} are computed in frame N+1.

The matrices ${}^{N+1}\mathcal{I}^{-1}_N$ and ${}^{N+1}\mathcal{I}^{-1}_N {}^{N+1}\hat{P}_N$ are constant in frame N+1 and hence can be precomputed. **The computation of $\mathcal{A}^{-1}\mathcal{E}$ is equivalent to the solution of**

$$\mathcal{A}\Omega = \mathcal{E} \tag{84}$$

for Ω. **This represents** the solution of a SPD block tridiagonal system for six right-hand side vectors which can be obtained in O(N) steps. Exploiting the sparse structure of $\mathcal{E}^T$, the computation of $\mathcal{E}^T\Omega$ can be reduced to

$$\Theta = \mathcal{E}^T\Omega = \mathcal{E}^T_N\Omega_N \in \mathbb{R}^{6\times6} \tag{85}$$

where $\Omega_N \in \mathbb{R}^{5\times6}$ is the Nth element of Ω. Λ^{-1} can be then computed from Eq. (82) by an addition of two 6×6 matrices. This implies that the cost of explicit computation of Λ^{-1} is of O(N). If the block LDL^T factorization of matrix $\mathcal{A}$ is already computed then Eq. (84) can be solved with a greater efficiency since only the forward elimination and back substitution steps need to be performed,

C. **Computation of $\mathcal{J}\dot{Q}_0$ and $\mathcal{J}^T F_{N+1}$**

The evaluation of $\mathcal{J}$ from the factorization given in Eq. (8), that is, by explicitly computing the product of its factors, results in a more conventional representation of $\mathcal{J}$ as

$$\mathcal{J} = [\hat{P}^T_{N+1,N} H_N, \ \hat{P}^T_{N+1,N-1} H_{N-1}, \ \ldots, \ \hat{P}^T_{N+1,1} H_1]$$

However, the computation of $\mathcal{J}\dot{Q}_0$ and $\mathcal{J}^T F_{N+1}$ in Eqs. (32) and (34) represent operator applications of $\mathcal{J}$ and $\mathcal{J}^T$ which can be performed without explicit computation of $\mathcal{J}$.

The evaluation of $\mathcal{J}\dot{Q}_0$ corresponds to the acceleration propagation in the N-E algorithm while setting Q to zero. It can be recursively computed as

$$\dot{V}''_{0i} = \hat{P}^T_i \dot{V}''_{0i-1} + H_i \dot{Q}_{0i} \qquad i = 1 \text{ to } N+1 \tag{86}$$

with $\dot{Q}_{0N+1} = 0$.

The evaluation of $\mathcal{J}^T F_{N+1}$ corresponds to the propagation of spatial forces among the rigidly connected links of the arm. Its recursive implementation is derived by setting $\dot{V}_i$ to zero in Eq. (12) and is given by

$$F_i = \hat{P}^T_i F_{i+1} \qquad i = N \text{ to } 1 \tag{87}$$

$$\cdot C'_i = H^T_i F_i \qquad i = N \text{ to } 1 \tag{88}$$

The computation of the linear recurrences in Eqs. (86)-(87) as well as that of Eq. (88) can be performed in O(N) steps. Since F_{N+1} is given in frame N+1 and $\dot{V}_{0N+1}$ (and hence $\dot{V}''_{0N+1}$ 1 also needs to be computed in this frame, it is more efficient to directly perform the computation of Eqs. (86)-(88) in frame N+1. This can be achieved by first projecting the vectors Z_i and P_i in frame N+1 and then performing the computation in Eqs. (86)-(88) in this frame.

D. Computational Efficiency of Serial Algorithm

For the solution procedure given in Table I, the application of the new factorization of $\mathcal{M}^{-1}$ and Λ^{-1}, as discussed above, results in a complexity of O(N) for computation of Steps II, III, and V. Step I requires the computation of the N-E algorithm which can be performed in O(N) steps and the computation of Step IV requires a flat cost independent of N. This implies an O(N) overall complexity of the serial algorithm. Therefore, the algorithm is asymptotically as fast as the previously proposed algorithms [5,6,8,9,10].

However, in terms of number of operations, this algorithm seems to be less efficient than the previous algorithms. The cost of the serial solution of forward dynamics of an open-chain arm by using the new factorization of $\mathcal{M}^{-1}$ is analyzed in detail in [13,14]. The analysis in [13,14] indicates that this factorization results in an O(N) algorithm which, in terms of total number of operations, is less efficient (by a factor of ≈3.4 for large N) than the best serial O(N) algorithms [15,17,18] for the problem.

For a single closed-chain arm, a somewhat improved efficiency can be expected. As discussed above, the most computationally-expensive part in implementing the factorization of $\mathcal{M}^{-1}$ is the block LDL^T factorization of matrix $\mathcal{A}$. However, for the two operator applications of $\mathcal{M}^{-1}$ as well as the computation of Λ^{-1} this factorization needs to be performed once. But, even by such an optimization, it seems unlikely that these new factorization can result in a highly competitive serial algorithm for the problem.

It should be pointed out that the manifestation of matrices Vi in the computation dictates a certain strategy for projection of the equations to achieve an optimal computational efficiency. This strategy is different from those usually proposed for computation of the inverse and forward dynamics of open- or closed-chain arm. As discussed in [13,14], any other projection strategy will significantly reduce the efficiency of serial and parallel algorithms resulting from these factorization.

VI. Parallel $O(\text{Log } N)$ Algorithms for Forward Dynamics of Single Closed-Chain Arm

A. Time and Processor Bounds in Computation

The efficiency of the new factorization of $\mathcal{M}^{-1}$ and Λ^{-1} for parallel solution of the problem can be assessed by examining the parallelism in the solution procedure of Table I while using these factorization. Here, parallel computation of Steps I-V by using O(N) processors is discussed.

Step I. By using the parallel algorithm in [37], the N-E algorithm can be computed in a time of O(Log N)+O(1) with O(N) processors. The vector addition in Eq. (30) can be performed in a fully decoupled fashion in a time of O(l).

Step II. The computation of the elements of matrix $\mathcal{A}$ from Eqs. (75)-(78) is fully decoupled and can be performed in O(1) steps, The matrix-vector multiplications in Eq. (81), which involve block diagonal and block bidiagonal matrices, as well as the vector addition also represent fully decoupled computations and can be performed in O(1) steps.

The block LDL^T factorization algorithm seems to be strictly serial and, in fact, there is no report on its parallelization. On the other hand, the BCR algorithm, while not being efficient for serial computation, is highly suitable for parallel computation, By using the parallel version of BCR algorithm [331, the block tridiagonal system in Eq. (81) can be solved in O(Log N)+O(1) steps with O(N) processors. This implies that the parallel implementation of Eq. (81) and hence the operator application of $\mathcal{M}^{-1}$ can be performed in O(Log N)+O(1) steps with O(N) processors.

The projection of vectors Z_i and P_i, as shown below, can be performed in O(Log N)+O(1) steps, The linear recurrence in Eq. (86) can be computed in O(Log N)+O(1) steps by using the Recursive Doubling Algorithm (RDA) [38]. The vector addition in Eq. (33) can be done in O(1) by using one processor.

Step III. By using the parallel version of BCR algorithm, the solution of the system in Eq. (84) for six right-hand side vectors can be computed in

$O(\text{Log } N)+O(1)$ steps with O(N) processors. The computation of $\mathcal{E}_N$ from Eq. (83), Θ from Eq. (85), and Λ^{-1} from Eq. (S2) involve simple matrix operations which can be performed in a time of O(1) by using one processor.

Step IV. As discussed in §III, regardless of the model of contact considered, the computation of Step IV involves simple matrix-vector operations which can be computed in a time of O(1) by using one processor.

Step V. With the vectors Z_i and P_i already projected in Step II, the linear recurrence in Eq. (87) can be computed in $O(\text{Log } N)+O(1)$ steps by using RDA and the computation in Eq. (88) is fully decoupled and can be done in O(l). As in Step II, the operator application of $\mathcal{M}$-l in Eq. (35) can be performed in a time of $O(\text{Log } N)+O(1)$ and the vector addition in Eq. (36) in a time of O(l).

It can be concluded the application of the new factorization of $\mathcal{M}$-l and Λ^{-1} enable the solution procedure of Table I to be computed in a time of O(Log N)+$O(1)$ by using O(N) processors. This indicates a both time- and processor-optimal parallel algorithm for the problem. In the following, a more detailed practical implementation of this parallel algorithm is discussed.

B. Parallel Solution of SPD Block Tridiagonal Matrix

The solution of the block tridiagonal systems represents the most computational ly-intensive part of the overall solution procedure. Therefore, a central issue that affects the efficiency of parallel computation is the choice of parallel algorithm for solution of the block tridiagonal system.

There are two variants of the BCR algorithm: the Odd-Even Reduction (OER) and Odd-Even Elimination (OEE) algorithms [33]. The OEE algorithm, while less efficient than the OER algorithm for serial computation, provides additional parallelism in both computation and communication while using the same number of processors and interconnection structure as for OER algorithm [33].

We consider the solution of a block tridiagonal system as

$$\mathcal{A}\Phi = \Psi \tag{89}$$

where $\Phi = \mathrm{col}\{\phi_i\}$ and $\Psi = \mathrm{col}\{\psi_i\}$, i = N to 1, with ϕ_i and ψ_i standing for subvectors or submatrices of appropriate size, The OEE algorithm is given as

```
For i = N to 1, Do
    $A_i^0 = A_i$, $B_i^0 = B_i$, and $\psi_i^0 = \psi_i$    (initialization)
End_Do
For j = 1 to $M = \lceil \mathrm{Log}_2 N \rceil$, Do
    For i = N to 1, Do
        $A_i^j = A_i^{j-1} - B_i^{j-1}(A_{i+2^{j-1}}^{j-1})^{-1}(B_i^{j-1})^T - (B_{i-2^{j-1}}^{j-1})^T(A_{i-2^{j-1}}^{j-1})^{-1}B_{i-2^{j-1}}^{j-1}$    (90)
        $B_i^j = -B_i^{j-1}(A_{i+2^{j-1}}^{j-1})^{-1}B_{i+2^{j-1}}^{j-1}$    (91)
    End_Do
End_Do
For j = 1 to $M = \lceil \mathrm{Log}_2 N \rceil$, Do
    For i = N to 1, Do
        $\psi_i^j = \psi_i^{j-1} - B_i^{j-1}(A_{i+2^{j-1}}^{j-1})^{-1}\psi_{i+2^{j-1}}^{j-1} - (B_{i-2^{j-1}}^{j-1})^T(A_{i-2^{j-1}}^{j-1})^{-1}\psi_{i-2^{j-1}}^{j-1}$    (92)
    End_Do
End_Do
For i = N to 1, Do
    Solve $A_i^M \phi_i = \psi_i^M$ for $\phi_i$    (93)
End_Do
```

where $\lceil x \rceil$ indicates the smallest integer greater than or equal to x. It should be noted that in Eqs. (90)-(91) it is more efficient to first compute the scalar ldl^T factorization of the dense submatrices rather than their explicit inverses. The multiplication of the inverse of a matrix by another matrix can be computed **as** the solution of a linear system with multiple right-hand sides.

The OEE algorithm can be regarded as a procedure for diagonalization of matrix $\mathcal{A}$ in which a sequence of transformations are applied to both sides of Eq. (89) resulting in a block diagonal system given by Eq. (93). In this sense, Eqs. (90)-(91) represent the diagonalization of matrix $\mathcal{A}$ while Eq. (92)

represents the updating of the right-hand side. Once matrix $\mathcal{A}$ is diagonalized, the solution of the linear systems required for operator applications of $\mathcal{M}^{-1}$ as well as the solution of the system in Eq. (S4) can be obtained by computing Eqs. (92)-(93) for the corresponding right-hand sides and by using the already computed submatrices A_i^j and B_i^j generated during diagonalization of matrix $\mathcal{A}$.

The parallel implementation of both the OER and OEE algorithms for scalar tridiagonal systems is straightforward. However, for block tridiagonal systems care should be taken to achieve the optimal efficiency. In fact, it seems that efficient implementation of either algorithms for block tridiagonal systems has received less attention in the literature. Note that, an implementation strategy represents a specific process-to-processor allocation scheme. To see this, consider the parallel implementation by using N processors, designated as PR_i, i = N to 1. A first possible (and more obvious) strategy for parallel implementation of the OEE algorithm is based on allocating the computation of A_i^j, B_i^j, ψ_i^j, and ϕ_i as well as all the intermediate terms in Eqs. (90)-(92) to processor PR_i. Note that, this strategy, which seems to be widely adopted in the literature for implementation of both the OER and OEE algorithms, is optimal for scalar tridiagonal systems.

We have developed a second strategy in which the terms A_i^j and ψ_i^j as well as all intermediate terms involving A_i^{j-1} are computed by PR_i. The two strategies lead to two different structures for the computations performed by each processor as well as the communication among processors. The impact of the two strategies on both computation and communication complexity of the algorithm is discussed in [39]. Here, suffice to mention that the second strategy, presented below, not only leads to a greater computational efficiency but, more importantly, it also provides a high degree of overlapping between the computation and the communication which can be exploited to reduce the communication overhead.

C. Strategy for Multilevel Parallel Computation

An imp ementation of the parallel O(Log N) algorithm with N processors follows d rectly from the analysis in Part A. However, the theoretical analysis n [13,14]- which is also supported by the practical implementation results in [40]- indicates that, for forward dynamics solution of an open-chain arm, the parallel computation of the Schur Complement factorization of $M^{a}$by N processors results in a parallel algorithm that, despite its asymptotic optimality, offers a limited speedup for the systems with small N. This is due to the large constant coefficients on the polynomial complexity of the resulting parallel algorithm. For forward dynamics solution of the closed-chain arm, given its greater computational cost, parallel computation by N processors, would result in even larger constant coefficients. Therefore, key to increasing the efficiency of parallel algorithm is to reduce the constant coefficients by exploiting a higher degree of parallelism in the computation through a multilevel approach and by using a larger number of processors.

A first possible strategy for multilevel parallel computation is to exploit fine-grain parallelism in various matrix-vector operations of the algorithm. However, this would require the implementation of the algorithm on special-purpose parallel architectures such as the one proposed in [28]. Here, we consider a second strategy based on a coarse-grain multilevel parallel computation with simple architectural requirements.

There are both algorithmic and architectural incentives for adopting this strategy, From an algorithmic standpoint, the application of the Schur Complement factorization of $\mathcal{M}^{-1}$ and A-* to the solution procedure of Table I results in a high degree of coarse grain parallelism. For example, the computation of Step I can be performed in parallel with the computation and diagonalization of matrix $\mathcal{A}$. Once matrix $\mathcal{A}$ is diagonalized, the computation of A^{-1} can be performed in parallel with the rest of the computations in Step I and Step II. Furthermore, the solution of the system in Eq. (84) for six right-side vectors essentially involves six decoupled processes which can be computed in parallel by using 6N processors.

From an architectural standpoint, this approach is motivated by the fact that, for small N, it is likely that the number of available processors of the target architecture be much greater than N. This clearly suggests to increase the efficiency of the algorithm by using more of the processors that would otherwise be idle.

Clearly, depending on the number of processors employed and hence the degree of parallelism exploited, a variety of multilevel parallel implementations of the algorithm can be considered. Furthermore, any implementation will also strongly depend on the features of the target architecture, e.g. , synchronization mechanism and processors' interconnection structure. In the following, we discuss an implementation of the algorithm by using 2N processors. As will be shown, the main advantages of this implementation are its simple synchronization and communication requirements.

D. Implementation of Multilevel Parallel O(Log N) Algorithm

For the implementation of the multilevel parallel algorithm, we consider two interconnected processor arrays, each with N processors. The processors are denoted as $PR_{k\,i}$, k = 1, 2, and i = 1 to N (Fig. 3 shows the two arrays for N = 8.) Each array is a SIMD architecture with a Shuffle Exchange augmented by Nearest Neighbor (SENN) interconnection structure (Fig 2). The SENN is optimal for the implementation of the parallel algorithm since it perfectly matches the inherent communication structure of different steps of the algorithm. In describing the computational steps of the algorithm, it is assumed that the constant kinematic and dynamic parameters of link i reside in the memory of processors $PR_{1\,i}$, and $PR_{2\,i}$.

The computation of the multilevel parallel algorithm involves an ordering which slightly differs from that of serial algorithm of Table I and it is performed according to the following steps.

Step I: Diagonalize Matrix $\mathcal{A}$ and Compute Λ^{-1} by Processor Array PR_{1i}

a. Projection and Computation of' Matrix $\mathcal{A}$

```
For i = 1 to N, Do_Parallel
1. Form $C(i,i+1)$ and $\mathcal{C}(i,i+1)$.
2. Compute ${}^{i}S_{i}$, $({}^{i}J_{i})^{-1}$, and $({}^{i}\mathcal{I}_{i})^{-1}$ from Eqs. (78), (77), and (76).
3. Send $({}^{i}\mathcal{I}_{i})^{-1}$ to $PR_{2,i}$.
4. Compute $B_i$ from Eqs. (79)-(80).
5. Compute $A_i$ from Eq. (75).
End_Do Parallel
```

b. Diagonalize Matrix $\mathcal{A}$

```
For j = 1 to M = $\lceil \mathrm{Log}_2 N \rceil$ , Do
    For i = 1 to N, Do Parallel (by all $PR_{1i}$'s)
      1. Compute $\ell d \ell^T$ factorization of $A_i^{j-1}$ [with $A_i^0 = A_i$).
      2. Solve $A_i^{j-1} C_i^{j-1} = B_i^{j-1}$ for $C_i^{j-1}$.
      3. Send $C_i^{j-1}$ to $PR_2$ ,.
      4. Compute $D_i^{j-1} = (B_i^{j-1})^T C_i^{j-1}$.
      5. Send $D_i^{j-1}$ to $PR_{1,i+2^{j-1}}$.
      6. Solve $A_i^{j-1} F_i^{j-1} = (B_{i-2^{j-1}}^{j-1})^T$ for $F_i^{j-1}$.
      7. Send $F_i^{j-1}$ to $PR_{2,i}$.
      8. Compute $G_i^{j-1} = (B_{i-2^{j-1}}^{j-1})^T (A_i^{j-1})^{-1} (B_{i-2^{j-1}}^{j-1})^T = (B_{i-2^{j-1}}^{j-1})^T F_i^{j-1}$.
      9. Send $G_i^{j-1}$ to $PR_{1,i-2^{j-1}}$.
     10. Compute $(B_{i-2^{j-1}}^{j})^T = (B_i^{j-1})^T (A_i^{j-1})^{-1} (B_{i-2^{j-1}}^{j-1})^T = (B^{j-1})^T F_i^{j-1}$.
     11. Send $(B_{i-2^{j-1}}^{j})^T$ to $PR_{1,i-2^{j-1}}$ and $PR_{1,i+2^{j-1}}$.
     12. Compute $A_i^j = A_i^{j-1} - D_{i-2^{j-1}}^{j-1} - G_{i+2^{j-1}}^{j-1}$.
    End_Do Parallel
End_Do
```

```
For i = 1 to N, Do Parallel (by all $PR_{1i}$'s)
13. Send $A_i^M$ to $PR_{2i}$.
End_Do Parallel
```

c. Compute Matrix Λ^{-1}

```
1. Initialization
For i = 1 to N, Do_Parallel (by all $PR_{1i}$)
    Set $\psi_i^0 = \mathcal{E}_i$
End_Do Parallel
2. Compute $\Psi^M$
For j = 1 to M = $\lceil Log_2 N \rceil$, Do
    For i = 1 to N, Do Parallel (by all $PR_{1i}$'s)
      i. Compute $E_i^{j-1} = (B_i^{j-1})^T (A_i^{j-1})^{-1} \psi_i^{j-1} = (C_i'^{j-1})^T \psi_i^{j-1}$.
     ii. Send $E_i^{j-1}$ to $PR_{1,i+2^{j-1}}$.
    iii. Compute $H_i^{j-1} = B_{i-2^{j-1}}^{j-1} (A_i^{j-1})^{-1} \psi_i^{j-1} = (F_i^{j-1})^T \psi_i^{j-1}$.
     iv. Send $H_i^{j-1}$ to $PR_{1,i-2^{j-1}}$.
      v. Compute $\psi_i^j = \psi_i^{j-1} - E_{i-2^{j-1}}^{j-1} - H_{i+2^{j-1}}^{j-1}$
    End_Do Parallel
End_Do
3. Compute $\Theta = \mathcal{E}_N^T \Omega = \mathcal{E}_N^T (A_N^M)^{-1} \psi_N^m$ by $PR_{1N}$.
4. Compute $\Lambda^{-1} = {}^{N+1}I_N - \Theta$ by $PR_{1N}$.
5. Send $\Lambda^{-1}$ to $PR_{2N}$ by $PR_{1N}$.
6. Wait (end of operation)
```

Step II. Compute $\mathcal{F}'_T$ and $\dot{V}_{0N+1}$ by Processor Array PR_{2i}

a. Projection

1. Form $C(i, i+1)$ in parallel for $i = N$ to 1.
2. Solve the linear recurrence $C(i+2, i) = C(i+2, i+1)C(i+1, i)$, $i = 1$ to $N-1$, in parallel by using RDA.

```
3, For i = 1 to N, Do_Parallel (by all $PR_{2\,i}$'s)
     i. Compute ${}^{N+1}Z_i = C(N+1, i)^i\, Z_i$
    ii. Compute ${}^{N+1}S_i = C(N+1, i+1)^{i+1}\, S_i$
   iii. Compute ${}^{N+1}P_i = C(N+1, i+1)^{i+1} P_i$
   End_Do Parallel
```

b. Compute $\mathcal{F}'_T$

```
1. Compute $b'(\theta, Q)$ by using the parallel algorithm in [37].
2. Compute $F'_{T\,i} = \tau_i - b'_i(\theta, Q)$, i = N to 1, in parallel.
3. Set $\mathcal{F}_G = \mathcal{F}'_T$.
4. Compute $\mathcal{X}^1 = \underline{\mathcal{I}}^{-1}\mathcal{R}\underline{\mathcal{H}}\mathcal{F}_G$.
5. Compute $\mathcal{X}^2 = \underline{\mathcal{W}}^T\mathcal{R}^T$.
6. Wait.
```

c. Compute $\dot{Q}_O$

```
1. Receive $A_i^M$ from $PR_2$ ,
2. Set $\psi_i^0 = \mathcal{X}_i^2$
3. Compute $\psi_i^m$ by repeating Step I.C.ii by processors $PR_{2\,i}$.
4. Solve Eq. (XX) in parallel for all $\phi_i$'s, i = N to 1, and set $\mathcal{X}^3 = \Phi$.
5. Compute $X^4 = \underline{\mathcal{I}}^{-1}\mathcal{R}\underline{\mathcal{W}}\mathcal{X}^3$
6. Compute $X^5 = \mathcal{X}^1 - X^4$
7. Compute $\dot{Q}_G = \underline{\mathcal{H}}^T\mathcal{R}^T\mathcal{X}^5$
8. Set $\dot{Q}_O = \dot{Q}_G$
```

d. Compute $\dot{V}_{ON+1}$

```
1. Solve the linear recurrence in Eq. (86) in parallel by using RDA.
2. Compute $\dot{V}_{ON+1}$ from Eq. (33) by processor $PR_{2\,N}$.
3. Wait.
```

Step III. Compute F_{N+1} by Processor $PR_{2\,N}$

```
1. Receive $\Lambda^{-1}$ from $PR_{1\,N}$.
2. Solve Eq. (29) for $K_C$ and compute $F_{N+1}$ from Eq. (24).
```

Step IV. Compute $\dot{Q}_O$ by Processor Array PR_{2i}

a. Compute $\mathcal{T}'$

1. Solve the linear recurrence in Eq. (S7) in parallel by using RDA.

2. Compute τ'_i from Eq. (88) in parallel for i = N to 1.

b. Compute $\dot{Q}_C$

1. Set $\mathcal{F}_G = \mathcal{T}'$, repeat Step 11. C.2-8, and set $\dot{Q}_C = \dot{Q}_G$.

2. Compute $\dot{Q}_i = \dot{Q}_{Oi} - \dot{Q}_{Ci}$ in parallel for i = N to 1.

E. **Performance of Multilevel Parallel Algorithm**

In order to appreciate the simple communication and synchronization requirements of the multilevel parallel algorithm, a brief discussion of its behavior is in order.

The processors PR_{1} ,' s are activated by receiving the data input θ and Q. The processors PR_{2i}'s are in turn activated by receiving the data input from processors PR_{1} ,'s, The activities of processor array $PR_{1,i}$ in computing Step I and those of processor array PR_{2i} in computing Step 11 are then performed in parallel and for the most part asynchronously. The cost of computing the matrices $({}^iI_i)^{-1}$ is much less than that of computing the vector $3'_T$. Therefore, the matrices $({}^iI_i)^{-1}$ are computed well before they are needed and hence their communication by processors PR_{2i}'s to processors $PR_{1,1}$'s can be performed asynchronously,

Both the computation and communication cost of diagonalization of matrix $\mathcal{A}$, as shown by theoretical analysis in [141 and practical implementation in [40], are greater than those of the evaluation of vector $\mathcal{F}'_T$. Therefore, upon computation of $\mathcal{X}^2$ (which requires a small amount of additional operations) the processors $PR_{2,}$, enter the wait state and become active by receiving the submatrices A_i^M. Note that, the communication of submatrices C_i^{j-1} and F_i^{j-1} can be performed asynchronously, i.e., they can be sent to processors PR_2, as soon as they are computed.

The computation of Λ^{-1} by processors PR_{1i} and $\dot{Q}_0$ and $\dot{V}_{ON+1}$ by processors PR_{2i} can be performed in parallel. It can be easily shown that both the computation and communication costs of the evaluation of A^{-1} are much greater than those of the evaluation of $\dot{Q}_0$ and $\dot{V}_{ON+1}$ since the former requires the computation Eq. (92) for six right-hand side vectors while the latter requires the same computation for only one right-hand side vector. Therefore, upon computation of $\dot{V}_{ON+1}$ the processors PR_{2i} enter the wait state and become active by receiving A^{-1} to perform the computation of Steps III and IV. Based on our discussion, it can be concluded that the computation of Step II can be totally overlapped with that Step I. As a result, the computation and communication costs of Step II do not contribute to the overall computation and communication cost of the algorithm.

The efficiency of our strategy for implementation of the OEE algorithm, in terms of communication overhead minimization, can be assessed by analyzing the diagonalization of matrix $\mathcal{A}$ in Step I.b and the computation of Ψ^M in Steps I.c In fact, as can be seen, any communication activity can be overlapped with its immediate computation activity. In most of emerging parallel architectures (both SIMD and MIMD) each node has one processor dedicated to computation and a second processor (e.g. , a DMA) dedicated to communication. "On these architectures, this overlapping of the communication with the computation can be exploited to significantly reduce the communication overhead.

It should **be** emphasized that the choice of the SIMD mode for parallel implementation of the algorithm is mainly motivated by the regularity in its computation. However, the algorithm has a rather large grain size since, particularly in Step I, each processor performs a matrix-vector operation or a series of such operations before communicating with other processors. This large grain size coupled with the possibility of overlapping the communication with the computation and the low level of communication activities, make the multilevel parallel algorithm highly suitable for implementation on MIMD architectures such as Hypercube.

In fact, our practical implementation of a parallel algorithm based on the Schur Complement factorization of $\mathcal{M}^{-1}$ for forward dynamics solution of open-chain arm on a MIMD Hypercube architecture [40] supports the efficiency of the proposed strategies for both implementation of the OEE algorithm and, particularly, coarse grain multilevel parallel computation. In particular, note that, the parallel algorithm in [37] for computation of $b'(\theta, Q)$ in Step II.b has a fine grain and its implementation on MIMD architectures leads to a very limited speedup which can degrade the overall performance of parallel computation. However, as was shown, with a multilevel parallel computation approach the computation of $b'(\theta, Q)$ can be fully overlapped with the rest of the computations. This eliminates the possibility of performance degradation due to the fine grain parallel computation of $b'(\theta, Q)$.

VII. Discussion and Conclusion

In this paper, we presented a new factorization technique for computation of $\mathcal{M}^{-1}$ and A^{-1}. This technique results in Schur Complement factorization of $\mathcal{M}^{-1}$ and A^{-1} and subsequently a new O(N) algorithm for forward dynamics solution of single close-chain arms. This O(N) algorithm is strictly efficient for parallel computation. That is, it is less efficient than previously proposed O(N) algorithms for serial solution of the problem. But, to our knowledge, it represents the first algorithm that can be fully parallelized, resulting in a both time- and processor-optimal parallel algorithm for the problem.

In addition to theoretical significance, the resulting parallel $O(\text{Log N})$ algorithm is also of practical importance. In fact, the algorithm achieves the time lower-bound in the computation while also providing a high degree of coarse grain parallelism which can be exploited with a rather simple communication and synchronization requirements

We did not evaluate the computational cost of either serial or parallel algorithms in terms of number of operations. As stated before, it is unlikely that our serial algorithm can become competitive with the previously proposed O(N) algorithms for the problem. The difficulty in assessing the relative efficiency of serial algorithm of this paper arise from the fact that the analysis of the O(N) algorithms in the literature is, for most part, limited to the asymptotic complexity. This makes the determination of the best serial algorithm, in terms of number of operations, highly difficult. Further analyses and comparative studies are needed to establish a better understanding of the relative efficiency of various algorithms.

The lack of such a knowledge on the most efficient serial algorithm also renders the analysis of the speedup of the parallel $O(\mathrm{Log}\ N)$ algorithms of this paper impossible. However, it is clear that the multilevel parallel computation of these algorithms can lead to a significant speedup in the computation, particularly, for highly redundant arms. In this sense, these multilevel parallel algorithms have immediate application for simulation of redundant arms such as those proposed for Space Station. In fact, the Space Station Remote Manipulator System (SSRMS) and the Special Purpose Dexterous Manipulator (SPDM) may have as many as 25 DOFS in total,

The analysis of this paper clearly indicates that the main application of these new factorization will most likely be limited to parallel computation of the problem. This suggests that further research works should be focused on devising more efficient strategies for their parallel implementation. In this sense, much can be learned from the results of the practical implementation of a multilevel parallel algorithm based on the Schur Complement factorization of M^{a} for open-chain system on Hypercube [401.

Acknowledgments

The *research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA).*

Appendix: Positive Definiteness of Matrix $\mathcal{A}$

Following theorem (see [36], pp. 140 for proof) is used in the analysis of positive definiteness of various matrices.

Theorem. If $A \epsilon \mathbb{R}^{n \times n}$ is positive definite and $X \epsilon \mathbb{R}^{n \times k}$ has rank k, then $B = X^T A X$ is also positive definite.

The matrix $\mathcal{M}^{-1} \epsilon \mathbb{R}^{N \times N}$ is the inverse of the mass matrix and is positive definite. Therefore, from the above theorem, the positive definiteness of matrix Λ^{-1}, given in Eq. (21), depends on the rank of Jacobian matrix $\mathcal{J} \epsilon \mathbb{R}^{6 \times N}$. If, due to the kinematic singularity, the rank of $\mathcal{J}$ becomes less than six then the matrix Λ^{-1} becomes singular. If Λ^{-1} is positive definite then it follows that the matrix $W^T_{N+1} \Lambda^{-1} W_{N+1}$, in Eq. (29), is also positive, definite since W_{N+1} is an orthogonal and hence a full rank matrix.

The matrix $\mathcal{A}$ is given as $\mathcal{A} = \mathcal{W}^T \mathcal{P}^T \mathcal{I}^{-1} \mathcal{P} \mathcal{W} \; \epsilon \mathbb{R}^{5N \times 5N}$. The positive definiteness of matrix $\mathcal{I}^{-1} \epsilon \mathbb{R}^{6NX6N}$ follows from posit ve definiteness of I_i and I_i^{-1}. It can be easily seen that the matrix $\mathcal{P} \epsilon \mathbb{R}^{6N \times 6N}$ has rank 6N. Hence, the matrix $\mathcal{P}^T \mathcal{I}^{-1} \mathcal{P} \epsilon \mathbb{R}^{6N \times 6N}$ is also positive definite. For a joint i with ni<6 DOF, the columns of the matrix $W_i \epsilon \mathbb{R}^{6 \times 6-ni}$ are orthogonal and hence it has rank 6 - ni. Therefore, the matrix $\mathcal{W} \epsilon \mathbb{R}^{6N \times 6N-m}$ has rank 6n-m where m denotes the total DOFS, that is, $m = \Sigma^n_{i=1}$ ni. Again, from the above theorem, it follows that the matrix $\mathcal{A}$ is also positive definite. The positive definiteness of matrix $\mathcal{C}$ follows by a similar reasoning and using the orthogonality of matrices H_i.

REFERENCES

1] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation, " IEEE J. Robotics and Automation, vol. RA-3(1), pp. 43-53, 19s7.

21 D.E. Orin and McGhee, "Dynamic Computer Simulation of Robotic Mechanisms, " in *Theory and Practice of Robot and Manipulators,* pp. 286-296, 1981.

[31 R. Featherstone, *Robot Dynamics Algorithms,* Ph.D. Dissertation, Univ. of Edinburgh, 1984,

[41 R. Featherstone, "The Dynamics of Rigid Body Systems with Multiple Concurrent Contacts, " Proc. 3rd Int. Symp. Robotics Research, pp. 189-196, 1986.

[5] R.H. Lathrop, ''Constraint (Closed-Loop) Robot Simulation by Local Constraint Propagation, " Proc. IEEE Int. Conf. Robotics and Automation, pp. 689-694, 1986.

[6] K.W. Lilly, *Efficient Dynamic Simulation of Multiple Chain Robotic Mechanism,* PhD Diss. The Ohio State Univ. , 1989.

[7] K.W. Lilly and D.E. Orin, "Efficient Dynamic Simulation of a Single Closed-Chain Manipulator, " Proc. IEEE Int. Conf. Robotics & Automation, pp. 210-215, Sacramento, CA, April 1991.

[81 K.W. Lilly and D.E. Orin, "Efficient O(N) Computation of the Operational Space Inertia Matrix, " Proc. IEEE Int. Conf. Robotics & Automation, pp. 1014-1019, Cincinnati, OH, May 1990.

[91 H. Brandle, R. Johanni, and M. Otter, "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion fo the Mass Matrix, " Proc. IFAC Int. Symp. on the Theory of Robots, 1986.

[10] G. Rodriguez, A. Jain, and K. Kreutz-Delgado, "A Spatial Operator Algebra for Manipulator Modeling and Control, " Int. J. Robotics Research, Vol. 10(4), pp. 371-381, Aug. 1991.

[11] S. McMillan, D.E. Orin, and P. Sadayappan, '’Toward Super-Real-Time Simulation of Robotic Mechanism Using a Parallel Integration Method, " IEEE Trans. Systems, Man, and Cybernetics, Vol. (XX) Jan,/Feb. 1992.

[12] A. Fijany and A.K. Bejczy, "Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics, " in *Advances in Control and Dynami(Systems, Vol. 40: Advances in Robotic Systems Dynamics and Control,* C.T. Leondes (Ed.), pp. 357-410, Academic Press, March 1991.

[13] A. Fijany, "Parallel O log N) Algorithms for Rigid Multibody Dynamics, "

JPL Engineering Memorandum, EM 343-92-125S, Aug. 1992.

[141 A. Fijany, I. Sharf, and G.M.T. D'Eleuterio, "Parallel O(Log N) Algorithms for Computation of Manipulator Forward Dynamics," Submitted to IEEE Trans. Robotics & Automation.

[151 R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertia, " Int. J. Robotics Research, Vol. 2(l); pp. 13-30, 1983.

[16] A. Jain, "Unified Formulation of Dynamics for Serial Rigid Multibody Systems, " J. Guidance, Control, and Dynamics, Vol. 14(3), pp. 531-542, May/June 1991.

[17 G. Rodriguez, ''Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics, " IEEE J. Robotics & Automation, Vol. RA-3(6), pp. 624-639, Dec. 19S7.

[18 G. Rodriguez and K. Kreutz-Delgado, "Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix, " IEEE Trans. Robotics & Automation, Vol. RA-8(1), pp. 65-76, Feb. 1992.

[191 A. Fijany and A.K. Bejczy, "Parallel Algorithms and Architecture for Computation of Robot Manipulator Forward Dynamics, " Proc. IEEE Int. Conf. Robotics & Automation, pp. 1156-1162, April 1991, Sacramento, CA.

[20 A. Fijany and R.E. Scheid, ''Fast Parallel Preconditioned Conjugate Gradient Algorithms for Robot Manipulator Dynamic Simulation, " To appear in J. Intelligent & Robotic Systems: Theory & Application, 1993. Also, in JPL Eng. Memo, EM 343-1196, Aug. 1991,

[211 H. Kasahara, H. Fujii, and M. Iwata, "Parallel Processing of Robot Motion Simulation, " Proc. 10th IFAC World Congress, July 1987.

[22] C.S.G. Lee and P.R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computation, " IEEE Trans. Syst. , Man, and Cybern. , Vol. 18(2), pp. 238-251, March/April 1988.

[231 I. Sharf, *Parallel Simulation Dynamics for* Open *Multibody* Chains, Ph.D. Diss. , Univ. of Toronto, Canada, Nov. 1990.

[241 1. Sharf and G.M.T. D'Eleuterio, "Parallel Simulation Dynamics for Rigid Multibody Chains, " proc. 12th Biennial ASME Conf. on Mechanical Vibration and Noise, Montreal, Canada, Sept. 1989.

[251 A. Fijany, "Parallel O(Log N) Algorithms for Open- and Closed-Chain Rigid Multibody Systems based on a new Mass Matrix Factorization Technique, " Proc. 5th NASA Workshop on Aerospace Computational Control, pp. 243-266, Santa Barbara, Aug. 1993.

[261 J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-Line Computational Scheme for Mechanical Manipulator, " ASME J. Dynamic Syst., Meas., Control, vol. 102, pp. 69-76, June 19S0.

[27] J. Angeles, "On Numerical Solution of Inverse Kinematic Problem, " Int. J Robotic Res., Vol. 4(2), pp. 21-37, 1985.

[281 A. Fijany and A.K. Bejczy, "ASPARC: An Algorithmically Specialized Parallel Architecture for Robotics Computations, " in *Parallel Computation Systems for Robotics: Algorithms and Architectures,* A. Fijany and A.K. Bejczy (Eds.), World Scientific Pub., 1992.

[29] P.C. Hughes and G.B. Sincarsin, "Dynamics of an Elastic Multibody Chain. Part B: Global Dynamics, " Int. J. Dynamics and Stability of Systems, Vol. 4(3&4), pp. 227-244, 1989.

[30] C.J. Damaren and G.M.T. D'Eleuterio, "On the Relationship between Discrete-Time Optimal Control and Recursive Dynamics for Elastic Multibody Chains, " Contemporary Mathematics, Vol. 97, pp. 61-77, 1989,

[311 R.W. Cottle, ''Manifestation of Schur Complement, " Linear Algebra and its Application, Vol. S, pp. 189-211, 1974.

[32] A. Fijany, "New Mass Matrix Factorization Techniques and Parallel O(Log N) Algorithms for Dynamic Simulation of Multiple Manipulator Systems, " In preparation.

[331 R.W. Hockney and C.R. Jesshope, *Parallel Computers,* Adam Hilger Ltd, 1981

[34] D. Heller, "Some Aspects of the Cyclic Reduction Algorithm for Block Tridiagonal Linear Systems, " SIAM J. Numer. Anal., Vol. 13(4), 1976.

[351 D.E. Heller, *Direct and Iterative Methods for Block Tridiagonal Linear Systems.* Ph.D. Diss., Carnegie-Mellon Univ., April 1977.

[36] G.H. Golub and C.F. Van Loan, *Matrix Computations,* 2nd Edition, The John Hopkins Univ. Press, 1989.

[371 C.S.G. Lee and P.R. Chang, "Efficient Parallel Algorithms for Robot Inverse Dynamics Computation, " IEEE Trans. Syst., Man, and Cybern. , vol. 16(4), pp. 532-542, July/August 1986.

[38] P.M. Kogge, "Parallel Solution of Recurrence Problems, " IBM J. Res. Dev., Vol. 18, pp. 138-148, March 1974.

[391 A. Fijany and N. Bagherzadeh, "Communication Efficient Cyclic Reduction Algorithms for Parallel Solution of Block Tridiagonal Systems, " Submitted to Information Processing Letters.

[40] A. Fijany, G. Kwan, and N. Bagherzadeh, "A Fast Algorithm for Parallel Computation of Multibody Dynamics on MIMD Parallel Architecture, " To be presented at Computing in Aerospace 9 Conf. , San Diego, CA, Oct. 1993.